SimOpt V1.0

User Manual

Edited by:
Vegard Aksnes, Marintek AS

December 17, 2012

Contents

1 Introduction

SimOpt is a general purpose optimization program accessible from SIMA. SimOpt utilizes a commercial optimization package called NLPQLP, developed by Prof. Klaus Schittkowski. For details regarding NLPQLP see Schittkowski (2009).

This user manual contains the following:

- A brief introduction to optimization theory.
- A short description of how to set up an optimization problem in SIMA using SimOpt.
- Description of input and output files.
- Description of communication between SIMA and SimOpt.

2 Basic optimization theory

2.1 Formulation of an optimization problem

A constrained optimization problem consists of:

- A function to minimize, often called an *objective function* or sometimes a *cost function*. We will stick to the term objective function.
- *Optimization* or *design variables*.
- *Constraints*.

In mathematical terms, this translates to

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to:} \;\; c_i(x) \geq 0, \; i \in \mathcal{I}. \tag{1}$$

In Eq. (1) the objective function is f, the vector x is the optimization variable and c_i with $i \in \mathcal{I}$ is a set of contraints. A point f x satisfying the constraints is called a *feasible point* and the set of all feasible points is called the *feasible set* Ω

$$\Omega = \{x \in \mathbb{R}^n \text{ such that } c_i(x) \geq 0 \text{ for all } i \in \mathcal{I}\}. \tag{2}$$

The feasible set is where a minimum to f is sought. Optimization problems are often formulated with equality constraints and upper and lower bounds on the optimization variables in addition to inequality constraints. This can be beneficial for certain solution techniques, but such constraints can always be rewritten as inequality constraints for the sake of simplicity.

A simple example is to minimize the material costs for a single mooring line, subject to certain rules and regulations. The optimization variables could then be length l and diameter d of the mooring line. Upper and lower bounds on l and d can together with limitations on axial tension form the constraints. The objective function could be the amount of material in the mooring line, i.e. $f(l, d) = ld^2$. In such an example the evaluation of the constraints involving line tension could typically involve running RIFLEX.

2.2 Gradient based methods for nonlinear problems

This subsection is to a large extent based on Nocedal and Wright (1999). There are two main classes of gradient based methods for nonlinear constrained problems:

- Penalty, barrier and augmented Lagrangian methods
- Sequential quadratic programming (SQP) methods

The idea of penalty, barrier and augmented Lagrangian methods is to replace a constrained optimization problem with a sequence of unconstrained problems by introducing penalty and barrier parameters. Penalty methods are most suitable for problems with only equality constraints. A quadratic penalty function penalty function $Q(x; \mu)$ is often defined as

$$Q(x; \mu) = f(x) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} c_i^2(x), \tag{3}$$

where $\mathcal{E}$ is the set of equality constraints and $\mu > 0$ is a penalty parameter. The second term of Eq. (3) is the penalty term which increases if the constraints are violated. The unconstrained problem is solved for a decreasing sequence of penalty parameters until a solution to the constrained problem is found.

Barrier methods are similar to penalty methods, but more suited for problems with only inequality constraints. A typical logarithmic barrier function can be defined as

$$P(x;\mu) = f(x) - \mu \sum_{i \in \mathcal{I}} \log c_i(x), \tag{4}$$

where $\mathcal{I}$ is the set of inequality constraints and $\mu > 0$ is a barrier parameter. The second term is the barrier term which blows up when x approaches the boundary of the feasible set. As for penalty methods, the procedure is to solve the unconstrained problem Eq. (4) for a decreasing sequence of barrier parameters μ.

Penalty and barrier methods need to be combined to handle general nonlinear problems need to be combined in order to handle both inequality and equality constraints. Some issues with penalty and barrier are:

- Contours of the penalty or barrier function are usually elongated when $\mu \to 0$ and makes it increasingly more difficult for unconstrained optimization routines to perform well.
- A "good" choice for starting point is needed when the penalty/barrier parameter μ is small.
- The Hessian of the penalty or barrier function may become ill-conditioned for small μ.

Augmented Lagrangian methods overcome some of the issues above. For an equality constrained problem, the augmented Lagrangian function may be defined as

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} c_i^2(x) \tag{5}$$

The idea is the same as for penalty and barrier methods, namely to solve and unconstrained problem instead of a constrained one, but details are omitted here. Augmented Lagrangian methods can also be formulated for general problems with both equality and inequality constraints.

The basic idea of sequential quadratic programming (SQP) methods is to formulate a quadratic subproblem in each iterate and apply the solution of the subproblem as a new search direction. If we consider the problem (eq:optprob), then a typical quadratic subproblem is

$$\min_p \frac{1}{2} p^T W_k p + \nabla f_k^T p, \tag{6}$$

subject to

$$\nabla c_i(x_k)^T p + c_i(x_k) \geq 0, \quad i \in \mathcal{I}, \tag{7}$$

where the constraints have been linearized. In Eq. (6) $W_k = W(x_k, \lambda_k)$ is the Hessian of the Lagrangian of the problem (1), i.e.

$$W(x,\lambda) = \nabla^2_{xx} \mathcal{L}(x,\lambda) = \nabla^2_{xx} \left(f(x) - \sum_{i \in \mathcal{L}} \lambda_i c_i(x) \right). \tag{8}$$

Quadratic programming algorithms, such as active-set methods, are applied to solve (6). See Nocedal and Wright (1999) for further details. Sequential quadratic programming methods are often efficient on both small and large scale problems.

3 Setting up an optimization problem

The optimization problem is formulated in SIMA. SIMA runs an external program SimOpt, which is an interface to the FORTRAN subroutine NLPQLP that performs the actual optimization.

Setting up an optimization problem in SIMA consists of the following three main steps:

1. Set up Riflex Task (could be any other "physics" task).
    - Define variables (must be double variables to enter an optimization problem) and refer to them at relevant places in the model.
    - Define condition set or space to be run.
    - Check that the initial system runs without errors.
2. Set up Postprocessor Task.
    - Specify the postprocessor.
    - Set up an objective (cost) function.
    - Set up constraints.
3. Set up Optimization Task.
    - Choose postprocessor.
    - Choose optimization variables from list of "free" variables in the Riflex Task. A "free" variable is a variable which is not part of the condition set or space. Specify initial values, upper and lower bounds, and increment for gradient evaluations.
    - Specify NLPQLP related parameters such as accuracy and maximum number of iterations.

One may now run optimization. Results will be written to a file named `opt.res` and to `nlpqlp.res`. The file `opt.res` is a condensed version of the detailed output file `nlpqlp.res` produced by NLPQLP.

References

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

K. Schittkowski. NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line Search - User's guide, 2009.

A Description of input and output files

SimOpt works with one input file `opt.inp` and two output files `opt.res` and `nlpqlp.res`. These files are described in this appendix. Further communication between the SimOpt and SIMA is piped through the standard stream, see Appendix B.

A.1 Input file - opt.inp

SimOpt reads a single input file `opt.inp`. This file contains static input parameters. Other quantities such as values of the optimization variables and the function values of the objective function and the constraints are communicated through the so-called standard stream (`stdin` and `stdout`) as described in Appendix B.

The parameters needed in the input file are listed in Table A.1. The order of the parameters must be as in Table A.1. An arbitrary number of lines with comments may be given. Comment lines should start with an apostrophe.

Table A.1: Description of the `opt.inp` input file. Parameters must be given in the order listed below. See the NLPQLP manual for further description of the parameters.

Parameter	Description	Input	Type	Default	Range
`L`	Number of parallel systems. Only `L`=1 is implemented so far.	Set by SIMA	Int	1	$\texttt{L} \geq 1$
`M`	Total number of constraints.	Set by SIMA	Int	-	$\texttt{M} \geq 0$
`ME`	Number of equality constraints.	Set by SIMA	Int	-	$\texttt{ME} \geq 0$
`N`	Number of optimization variables.	Set by SIMA	Int	-	$\texttt{N} \geq 1$
`ACC`	Desired final accuracy. Should not be much smaller than the accuracy by which the gradients are computed.	User input	Float	1.0E-10	$\texttt{ACC} > 0$
`ACCQP`	Tolerance needed for the QP solver to perform several tests, for example whether optimality conditions are satisfied or whether a number is considered as zero or not.	Set by SIMA	Float	1.0E-12	$\texttt{ACCQP} > 0$
`STPMIN`	Minimum step length in case of `L`>1. Recommended is any value in the order of the accuracy by which the functions are computed.	Set by SIMA	Float	1.0E-12	$\texttt{STPMIN} > 0$
`MAXFUN`	Maximum number of function calls during line search. Dummy if `L`>1.	User input	Int	20	$0 < \texttt{MAXFUN} < 50$
`MAXIT`	Maximum number of outer iterations, where one iteration corresponds to one formulation and solution of the quadratic programming subproblem, or, alternatively one evaluation of the gradients.	User input	Int	50	$0 < \texttt{MAXIT} < 200$
`MAXNM`	Stack size for storing merit function values at previous iterations for non-monotone line search. If `MAXNM`=0, monotone line search is performed.	Set by SIMA	Int	10	$0 \leq \texttt{MAXNM} < 50$
`RHOB`	Parameter for performing a restart in case of `IFAIL`=2 by setting the BFGS-update matrix to `RHOB`*I, where I denotes the identity matrix. The number of restarts is bounded by `MAXFUN`. A value greater than one is recommended, but 0.0 is used in NLPQLP examples!	Set by SIMA	Float	0.0	$\texttt{RHOB} = 0.0$
`IPRINT`	Specification of the desired output level. `IPRINT`=4 gives "all" output.	Set by SIMA	Int	4	$\texttt{IPRINT} \in \{0, 1, 2, 3, 4\}$
`MODE`	Specified the desired version of NLPQLP. At present, only `MODE`=0 is allowed.	Set by SIMA	Int	0	$\texttt{MODE} = 0$

A.2 Output file - opt.res

The output file `opt.res` contains a header with the program name and the version, followed by the parameters read from `opt.inp`. The following is then written to file before each call to `NLPQLP`:

- Current values of the optimization variables
- Lower bounds for the optimization variables
- Upper bounds for the optimization variables
- Current function value of the objective function
- Current function values of the constraints
- Current gradient of the objective function
- Current gradients of the constraints
- Value of the IFAIL, which is a control flag for the `NLPQLP` subroutine

The file `opt.res` is parsed by SIMA for monitoring of the optimization process and for presentation of final results.

An example of `opt.res` is given below.

```
SimOpt - Optimization Software by MARINTEK AS
' Version
 1.0
'L
   1
'M
   2
'ME
   0
'N
   3
'ACC
 0.1000000000D-09
'ACCQP
 0.1000000000D-11
'STPMIN
 0.1000000000D-09
'MAXFUN
  10
'MAXIT
 100
'MAXNM
   0
'RHOB
 0.0000000000D+00
'IPRINT
   4
'MODE
   0
ITERATION
'Optimization variables
 0.1000000000D+02
 0.1000000000D+02
 0.1000000000D+02
'Optimization variables - Lower bounds
 0.0000000000D+00
 0.0000000000D+00
 0.0000000000D+00
'Optimization variables - Upper bounds
 0.4200000000D+02
 0.4200000000D+02
 0.4200000000D+02
'Objective function
-0.1000000000D+04
'Constraints
 0.5000000000D+02
 0.2200000000D+02
'Gradient of objective function
-0.1000000000D+03
-0.1000000000D+03
-0.1000000000D+03
'Gradients of constraints
 0.1000000000D+01
 0.2000000000D+01
```

```
 0.2000000000D+01
-0.1000000000D+01
-0.2000000000D+01
-0.2000000000D+01
'IFAIL
   0
ITERATION
'Optimization variables
 0.4200000000D+02
 0.7500000000D+01
 0.7500000000D+01
'Optimization variables - Lower bounds
 0.0000000000D+00
 0.0000000000D+00
 0.0000000000D+00
'Optimization variables - Upper bounds
 0.4200000000D+02
 0.4200000000D+02
 0.4200000000D+02
'Objective function
-0.2362500000D+04
'Constraints
 0.7200000000D+02
 0.0000000000D+00
'Gradient of objective function
-0.1000000000D+03
-0.1000000000D+03
-0.1000000000D+03
'Gradients of constraints
 0.1000000000D+01
 0.2000000000D+01
 0.2000000000D+01
-0.1000000000D+01
-0.2000000000D+01
-0.2000000000D+01
'IFAIL
  -1
   .
   .
   .
   and so on...
```

A.3 Output file - nlpqlp.res

The output file `nlpqlp.res` contains direct output from the `NLPQLP` subroutine. An example of `nlpqlp.res` is given below. SIMA does not use this file directly, but the file is available from the Optimization Task in SIMA and may be studied by the user if needed.

```
 ------------------------------------------------------------------
 START OF THE SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
 ------------------------------------------------------------------

 Parameters:
    N      =       3
    M      =       2
    ME     =       0
    MODE   =       0
    ACC    =   0.1000D-09
    ACCQP  =   0.1000D-11
    STPMIN =   0.1000D-09
    RHO    =   0.0000D+00
    MAXFUN =      10
    MAXNM  =       0
    MAXIT  =     100
    IPRINT =       4


 Iteration       0

    Function value:                F(X)  = -0.10000000D+04
    Variable:                      X     =
       0.10000000D+02  0.10000000D+02  0.10000000D+02
    Multipliers for lower bounds:  U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00
    Multipliers for upper bounds:  U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00
    Constraints:                   G(X)  =
       0.50000000D+02  0.22000000D+02
    Multipliers for constraints:   U     =
       0.00000000D+00  0.00000000D+00
    Sum of constraint violations:  SCV   = 0.0000D+00
    Number of active constraints:  NAC   =          2
    KKT optimality condition:      KKT   = 0.4364D+04
    Norm of Lagrangian gradient:   NLG   = 0.6550D+02
    QN-update denominator:         DBD   = 0.1036D+04
    Penalty parameter:  R =
       0.10000000D-02  0.10126506D+02  0.10000000D-02  0.10000000D-02
       0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
    Sufficient decrease reference value: DLP =-0.2700D+04
    Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.24E+04
    Line search successful after one step:  ALPHA = 1


 Iteration       1

    Function value:                F(X)  = -0.23625000D+04
    Variable:                      X     =
       0.42000000D+02  0.75000000D+01  0.75000000D+01
    Multipliers for lower bounds:  U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00
    Multipliers for upper bounds:  U     =
       0.16750000D+02  0.00000000D+00  0.00000000D+00
```

```
   Constraints:                   G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
      0.00000000D+00  0.51250000D+02
   Sum of constraint violations:  SCV   = 0.0000D+00
   Number of active constraints:  NAC   =          1
   KKT optimality condition:      KKT   = 0.1102D+04
   Norm of Lagrangian gradient:   NLG   = 0.1449D+03
   QN-update denominator:         DBD   = 0.1102D+04
   Penalty parameter:  R =
      0.10000000D-02  0.63644343D+01  0.10000000D-02  0.10000000D-02
      0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
   Sufficient decrease reference value: DLP =-0.1284D+04
   Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.33E+04
   Line search successful after one step:  ALPHA = 1


Iteration      2

   Function value:                F(X)  = -0.32507304D+04
   Variable:                      X     =
      0.31114571D+02  0.10221357D+02  0.10221357D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraints:                   G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
      0.00000000D+00  0.85064595D+02
   Sum of constraint violations:  SCV   = 0.0000D+00
   Number of active constraints:  NAC   =          1
   KKT optimality condition:      KKT   = 0.6932D+03
   Norm of Lagrangian gradient:   NLG   = 0.5631D+02
   QN-update denominator:         DBD   = 0.6932D+03
   Penalty parameter:  R =
      0.10000000D-02  0.33037756D+02  0.10000000D-02  0.10000000D-02
      0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
   Sufficient decrease reference value: DLP =-0.6932D+03
   Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.33E+04
   Line search  2: ALPHA = 0.54D+00, merit function FCT = -0.35E+04
   Line search successful after  2 steps:  ALPHA =   0.5417D+00


Iteration      3

   Function value:                F(X)  = -0.34557650D+04
   Variable:                      X     =
      0.24228877D+02  0.11942781D+02  0.11942781D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraints:                   G(X)  =
```

```
   0.72000000D+02  0.00000000D+00
Multipliers for constraints:   U     =
   0.00000000D+00  0.12608654D+03
Sum of constraint violations:  SCV   = 0.0000D+00
Number of active constraints:  NAC   =          1
KKT optimality condition:      KKT   = 0.5513D+00
Norm of Lagrangian gradient:   NLG   = 0.1490D+01
QN-update denominator:         DBD   = 0.5513D+00
Penalty parameter:  R =
   0.10000000D-02  0.83854021D+03  0.10000000D-02  0.10000000D-02
   0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
Sufficient decrease reference value: DLP =-0.5513D+00
Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.35E+04
Line search successful after one step:  ALPHA = 1


Iteration      4

   Function value:               F(X)  = -0.34559928D+04
   Variable:                     X     =
      0.23959948D+02  0.12010013D+02  0.12010013D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraints:                  G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
      0.00000000D+00  0.14412015D+03
   Sum of constraint violations:  SCV   = 0.0000D+00
   Number of active constraints:  NAC   =          1
   KKT optimality condition:      KKT   = 0.1452D-01
   Norm of Lagrangian gradient:   NLG   = 0.2410D+00
   QN-update denominator:         DBD   = 0.1452D-01
   Penalty parameter:  R =
      0.10000000D-02  0.14478779D+03  0.10000000D-02  0.10000000D-02
      0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
   Sufficient decrease reference value: DLP =-0.1452D-01
   Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.35E+04
   Line search successful after one step:  ALPHA = 1


Iteration      5

   Function value:               F(X)  = -0.34560000D+04
   Variable:                     X     =
      0.24000192D+02  0.11999952D+02  0.11999952D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraints:                  G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
```

```
      0.00000000D+00  0.14399942D+03
   Sum of constraint violations:  SCV   = 0.0000D+00
   Number of active constraints:  NAC   =          1
   KKT optimality condition:      KKT   = 0.3306D-06
   Norm of Lagrangian gradient:   NLG   = 0.1151D-02
   QN-update denominator:         DBD   = 0.3306D-06
   Penalty parameter:  R =
      0.10000000D-02  0.72196680D+02  0.10000000D-02  0.10000000D-02
      0.10000000D-02  0.10816867D+01  0.10000000D-02  0.10000000D-02
   Sufficient decrease reference value: DLP =-0.3306D-06
   Line search  1: ALPHA = 0.10D+01, merit function FCT = -0.35E+04
   Line search successful after one step:  ALPHA = 1


Iteration      6

   Function value:               F(X)  = -0.34560000D+04
   Variable:                     X     =
      0.24000000D+02  0.12000000D+02  0.12000000D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraints:                  G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
      0.00000000D+00  0.14400000D+03
   Sum of constraint violations:  SCV   = 0.0000D+00
   Number of active constraints:  NAC   =          1
   KKT optimality condition:      KKT   = 0.2300D-12
   Norm of Lagrangian gradient:   NLG   = 0.9596D-06


   Objective function value:      F(X)  = -0.34560000D+04
   Solution values:               X     =
      0.24000000D+02  0.12000000D+02  0.12000000D+02
   Distances from lower bounds:   X-XL  =
      0.24000000D+02  0.12000000D+02  0.12000000D+02
   Distances from upper bounds:   XU-X  =
      0.18000000D+02  0.30000000D+02  0.30000000D+02
   Multipliers for lower bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Multipliers for upper bounds:  U     =
      0.00000000D+00  0.00000000D+00  0.00000000D+00
   Constraint values:             G(X)  =
      0.72000000D+02  0.00000000D+00
   Multipliers for constraints:   U     =
      0.00000000D+00  0.14400000D+03
   Number of function calls:      NFUNC =      8
   Number of gradient calls:      NGRAD =      7
   Number of calls of QP solver:  NQL   =      7
```

B Communication between SimOpt and SIMA

Temporary description:

- Start the program...
- SimOpt reads a file "opt.inp" with "static" parameters for the optimization routine
- SimOpt reads initial values for the optimization variables from stdin (one value at a time)
- SimOpt reads lower bounds for the optimization variables from stdin (one value at a time)
- SimOpt reads upper bounds for the optimization variables from stdin (one value at a time)
- SimOpt reads the current value of the objective (cost) function from stdin (one value)
- SimOpt reads the current values of the constraint functions from stdin (one value at a time)
- SimOpt reads the current values of the gradient of the objective (cost) function from stdin (one value at a time)
- SimOpt reads the current values of the gradients of the constraint functions from stdin (one value at a time). The values are given in the following order:

$$\frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_2}, ..., \frac{\partial g_1}{\partial x_n}, \frac{\partial g_2}{\partial x_1}, \frac{\partial g_2}{\partial x_2}, ..., \frac{\partial g_2}{\partial x_n}, ..., \frac{\partial g_m}{\partial x_1}, \frac{\partial g_m}{\partial x_2}, ..., \frac{\partial g_m}{\partial x_n}$$

- SimOpt calls NLPQLP optimization routine
- SimOpt writes the value of IFAIL to stdout.
- SimOpt writes the new values for the optimization variables to stdout
- SimOpt checks the value of IFAIL:
  - If IFAIL=0 => optimization finished successfully
  - If IFAIL>0 => optimization failed. Possible errors are detailed in the source code and in the NLPQLP manual
  - If IFAIL=-1 => SimOpt reads the values of the objective function and contraints as above and calls NLPQLP again
  - If IFAIL=-2 => SimOpt reads the values of the gradients of the objective function and contraints as above and calls NLPQLP again
- SimOpt continues until IFAIL>=0